2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Real-Time Unified Trajectory Planning and Optimal Control for Urban Autonomous Driving Under Static and Dynamic Obstacle Constraints

Rowan Dempster*, Mohammad Al-Sharman, Derek Rayside, and William Melek

***Abstract*— Trajectory planning and control have historically been separated into two modules in automated driving stacks. Trajectory planning focuses on higher-level tasks like avoiding obstacles and staying on the road surface, whereas the controller tries its best to follow an ever changing reference trajectory. We argue that this separation is (1) flawed due to the mismatch between planned trajectories and what the controller can feasibly execute, and (2) unnecessary due to the flexibility of the model predictive control (MPC) paradigm. Instead, in this paper, we present a unified MPC-based trajectory planning and control scheme that guarantees feasibility with respect to road boundaries, the static and dynamic environment, and enforces passenger comfort constraints. The scheme is evaluated rigorously in a variety of scenarios focused on proving the effectiveness of the optimal control problem (OCP) design and real-time solution methods. The prototype code will be released at github.com/WATonomous/control.**

I. INTRODUCTION

In the SAE AutoDrive Challenge [1], the Society of Automotive Engineers (SAE) and General Motors (GM) set forth guidelines for vehicle dynamics metrics that should be obeyed by autonomous vehicles (AVs) to ensure comfortable and safe urban driving. These guidelines include limits on longitudinal acceleration and jerk, as well as lateral acceleration in the vehicle's body frame. The guidelines are to be adhered to as the AV accomplishes the dynamic driving task (DDT). Generally, the DDT can be seen as progressing towards a goal state in a feasible and efficient manner. In the context of urban driving, feasible means without collisions and while obeying traffic rules, and efficient means operating near the speed limit.

A. *Motivation*

There are several papers in the literature that address the real-time obstacle avoidance problem using optimal control techniques. However, all have drawbacks that make them ill-suited for the problem presented above. The authors of [2] apply a nonlinear MPC method, similar to the one presented in Section III, to the obstacle avoidance problem. However, the controller must be instructed by the perception system on which direction to avoid the obstacle (left or right). This assumption is unsatisfactory because the directional decision in itself needs to consider the dynamics of the vehicle to know if a maneuver to the left or right of the obstacle is most optimal. Our proposed method makes no such assumption.

R. Dempster, M. Al-Sharman, and D. Rayside are with the Department of Electrical and Computer Engineering, University of Waterloo. W. Melek is with the Department of Mechanical and Mechatronics Engineering, University of Waterloo. All authors are with the WATonomous lab, University of Waterloo, ON, N2L3G1, Canada. *Corresponding author: r2dempst@watonomous.ca

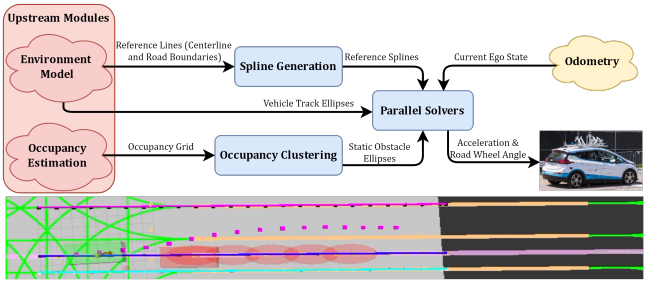

Fig. 1: The motion planning data pipeline discussed in Section II.

The authors of [3] take an interesting approach towards integrated obstacle avoidance by representing obstacles as a potential field in the cost formulation. However, this paper, as well as [4], [5], and [6] have the issue that the avoidance maneuver controller (which they separate from the lower level actuator controller) is based on a particle representation of the vehicle. Therefore, these approaches have the same issue as using a higher level planner: the planned states generated by the lower fidelity planning model may not be dynamically feasible and therefore are unsafe. Our approach uses a dynamics model for planning and thus guarantees dynamically feasible.

In [7], a cubic polynomial is utilized to describe the lateral deviation from the reference line. Similarly, the authors of [8] and [9] use a sigmoid function. However, none of these papers provide an analysis of why these functions were chosen in the context of their OCP objective function. In contrast, the scheme proposed here avoids obstacles in a manner that is consistent with its OCP objective function because the obstacles themselves are part of the OCP formulation.

In [10], an MPC-based technique for short-term path planning among multiple moving objects is presented. However, the work uses a linearized bicycle model (we use the nonlinear bicycle model), which is inaccurate during avoidance maneuvers that require large road wheel angles. Furthermore, obstacle information is assumed known at system start-up and the evaluation scenario is a straight road. In contrast, our system measures obstacles online via a LiDAR and performs well during turns.

B. *Contributions*

This paper addresses the issues presented above by introducing a novel OCP formulation of the DDT and an accompanying MPC solution. The main contributions in this regard are:

1) The system model is designed to align closely with physical platforms and to allow for computation of comfort metrics and constraints, enabling the OCP

to be formulated to abide by the SAE guidelines for passenger comfort.

2) Constraints are added to the OCP formulation that guarantee feasibility of control actions with respect to road boundaries, static obstacles, and dynamic vehicles. In this way, trajectory planning can be executed simultaneously within the controller, and the final control actions are guaranteed to be dynamically feasible.
3) The controller operates in real-time by employing a novel parallel-solver method that allows for a warm-started "online" solver, while employing a parallel "exploration" solver which serves to break out of local warm-starting local minima.

The remainder of this paper is organized as follows: Section II presents necessary background information on routing, reference line parameterization, and obstacle representations. Section III presents the OCP formulation, including the system model, objective function, road boundary constraints, and obstacle constraints. Section IV discusses the MPC solution to the OCP, including how the OCP is expressed as a nonlinear program (NLP), and the novel parallel-solver method for real-time performance. Experimental results are then presented and analyzed in Section V. Conclusions are discussed in Section VI.

II. BACKGROUND

In order to properly understand the motion planning scheme in which the proposed controller fits, some background information is necessary. We first note the information given to the motion planning scheme: (1) a high definition (HD) lane map, (2) the desired goal position on the map, (3) a discretized occupancy grid representation of the static environment, (4) non-ego dynamic vehicle tracks that describe the position and longitudinal velocity of the vehicle.

A. *Reference Spline Creation*

In order to transform this information into reference signals that the controller can follow, we first need to find a lane-level route that describes how the vehicle can proceed from its current state to the goal state. This step is called global planning and is done using Dijkstra's search algorithm over the routing graph constructed from the HD map [11].

However, a sequence of lanes is not yet an admissible reference signal for the controller. In order to generate a continuous and differentiable reference signal, a spline is fit to the centers of the lanes that make up the global route. In brief, CasADi's Interior Point Optimizer (IPOPT) [12] is used to find the spline coefficients that minimize squared error to the lane centers, see [8] for details.

An important note is that a spline has a limited capacity to express the complex lane geometry which may appear over a long route, incurring high squared error. To solve this problem, a sliding window is applied over the entire lane route, allowing the spline to accurately capture the simpler topology of the lanes in the local window.

The same method is applied to generate splines for the left and right boundaries of the driveable surface. The lines that describes the driveable surface are determined using the routing rules stored in the HD map.

Algorithm 1 Region Grow

Input: occupancy *grid*, row *r*, col *c*, radius *rad*, visited 2D array *visited*

Output: *[N, 2] array*, *N* is the number of cells in cluster

```
Initialize seedList ← [(r,c)]
Initialize cluster ← [(r,c)]
Initialize neighbors ← [n for c in product(range(-rad, rad + 1), repeat=2]) if n[0] != 0 or c[1] != 0]
while len(seedList) > 0 do
    Initialize currCell ← seedList.pop(0)
    visited[currCell[0]][currCell[1]] = 1
    for neigh ∈ neighbors do
        Initialize tR ← currCell[0] + neigh[0]
        Initialize tC ← currCell[1] + neigh[1]
        if grid[tR][tC] == 1 and visited[tR][tC] == 0 then
            seedList.append((tR, tC))
            cluster.append((tR, tC))
        end if
    end for
end while
return cluster
```

B. *Occupied Space Representations*

Aside from the centerline and road boundary splines, a reference for occupied space over time is also necessary to allow the controller to plan feasible trajectories. Two inputs are used to calculate this reference: An occupancy grid, and localized vehicle tracks from the tracker and environment model. The goal is to have a set of obstacles, and for each obstacle have a state trajectory that covers the MPC predictive horizon. The obstacle state representation used is a 2D ellipse, which has 5 degrees of freedom: X and Y of the centroid, RX (longitudinal radius), RY (lateral radius) and θ (yaw). To transform an occupancy grid into a set of obstacle trajectories, the grid is first filtered to only contain information of obstacles that are on the driveable surface. Then, a region growing algorithm is run with a neighborhood radius of four cells to cluster the obstacles in the grid (see Alg. 1). Lastly, the Minimum Volume Enclosing Ellipsoid (MVEE) approximate iterative algorithm [13] [14] is used to fit an ellipse to each cluster found by the region growing algorithm (see Alg. 2). Since the occupancy grid contains information about the static portions of the environment only, the occupancy ellipses are the same for each step in the MPC horizon.

To generate the ellipse trajectories for the dynamic vehicle tracks, the environment model module [11] first localizes each track in the lanelet it geometrically occupies. Then, based on the estimated longitudinal velocity (assumed constant) from the tracker, and the geometry of the upcoming lanelets, the ellipses are generated following Alg. 3. This simple prediction scheme can be expanded to a more complex learning based technique, but this simple method

Algorithm 2 Grid To Ellipses

Input: occupancy *grid*, neighbor radius *rad*
Output: *[N, 5] array*, *N* is the number of ellipses in grid

```
Initialize ells ← []
Initialize visited ← zeros_like(grid)
for r in range(grid.height) do
    for c in range(grid.width) do
        if visited[r][c] == 0 and grid[r][c] == 1 then
            cluster = RegionGrow(grid, r, c, rad, visited)
            C, rx, ry, theta = MVEE(cluster)
            ells.append(C.x, C.y, rx, ry, theta)
        end if
    end for
end for
return ells
```

Algorithm 3 Predicted Dynamic Vehicle Trajectory

Input: *envModel*, *track*, MPC step *S* (in seconds), MPC horizon *M*
Output: *[M, 5] array*, vehicle ellipse trajectory prediction

```
path = envModel.localizeVehicle(track)
startDist = toArcCoordinates(path, track.pos)
traj = []
for i in range(M) do
    lonDist = startDist + i * S * track.lonVel
    p1 = interpPointAtDistance(path, lonDist)
    p2 = interpPointAtDistance(path, lonDist + 0.1)
    theta = atan2(p2.y - p1.y, p2.x - p1.x)
    traj.append({ p1.x, p1.y, track.rx, track.ry, theta })
end for
return traj
```

suffices for controller design.

In summary, the information given to the controller is (as shown in Fig. 1):

1) A spline (denoted S_{ref}) that describes a local window of the global route.
2) Two splines (denoted S_{left} and S_{right}) which describe a local left and right drivable surface boundary.
3) A set of predicted trajectory ellipses (denoted O, where O_1 is the set of obstacles at step 1, and $O_{1,1}$ is the prediction for obstacle 1 at the 1st step in the MPC horizon).

III. OCP FORMULATION

This section describes the plant system to be controlled, the predictive model used in the proposed MPC controller, the objective function, and the constraints.

A. System and Predictive Models

Input Variables: The plant speed is controlled by applying a longitudinal acceleration command, and yaw is controlled via commanded road wheel angle. Additional, an input variable referred to as *path progress* u_ξ controls how far along the reference spline the next state will progress. The usage of this path progress input in the context of path following is explained below.

State Variables: The state vector was chosen in order to maintain the vehicle dynamic metrics that are necessary to compute the constraints presented below. Specifically, the state vector maintains the vehicle's inertial pose tuple $[x_X, x_Y, x_\psi]^T$ as well as longitudinal and forward body frame velocities (denoted x_{v_x}, and $x_{v_{fwd}}$, respectively). Additionally, a state variable referred to as the *path integral* x_Ξ is defined as the integration of the path progress input mentioned above. The path integral variable keeps track of how far along the reference route the ego is, and its functionality in terms of the objective function is explained below.

Output Variables: The system output variables are the odometry solutions generated by the Inertial Navigation System (INS). The INS directly measures inertial pose, as well as longitudinal and lateral velocities and accelerations in the body frame. The current path integral is estimated using a discretized linear search over the reference spline's parameter range.

Predictive Model: To describe the evolution of the system over time, a nonlinear model is necessary due to the nonlinear behavior of vehicular systems under large road wheel angles [15]. In an urban driving setting, maneuvers that require large road wheel angles are common. This is in contrast with a highway driving setting that may only require a few degrees of road wheel angle and hence operate inside of a linear dynamics range. To this end, a nonlinear kinematic bicycle model is employed, as illustrated in [16], [17] and described by Eqs. 1, 2:

$$\dot{x}_X = x_{v_{fwd}} cos(x_\psi + \beta) \tag{1a}$$

$$\dot{x}_Y = x_{v_{fwd}} sin(x_\psi + \beta) \tag{1b}$$

$$\dot{x}_\psi = \frac{x_{v_{fwd}} cos(\beta) tan(u_\delta)}{l_f + l_r} \tag{1c}$$

$$\dot{x}_{v_x} = u_a \tag{1d}$$

$$\dot{x}_\Xi = u_\xi \tag{1e}$$

where

$$\beta = \arctan(\frac{l_r tan(u_\delta)}{l_f + l_r}) \tag{2}$$

is the slip angle of the vehicle at its center of gravity (CoG).

In summary, a nonlinear kinematic bicycle model was chosen as it is a simple predictive model that captures the nonlinear system dynamics for low-speed urban navigation [15], and thus is the best trade-off between NLP computation time and accurate prediction.

B. Objective Function

The terms included in the OCP objective function are:

1) Reference position error. Calculated as squared error between x_X, x_Y and the reference spline at arclength x_Ξ.
2) Distance of path parameter to route completion. Calculated as $(1 - x_\Xi)$.

Giving the final objective function:

$$J = \sum_{k=1}^{N} \quad \left[\begin{bmatrix} x_X^k \\ x_Y^k \end{bmatrix} - \begin{bmatrix} x_{ref}^k \\ y_{ref}^k \end{bmatrix} \right]_2 + Q(1 - x_\Xi^k) \tag{3}$$

where Q is a positive number. Thus, the higher the Q value, the more incentivized the ego will be to complete the route, even at the cost of deviating from the reference route if needed. The outcome of the tuning process of Q was two separate weighting settings, one for nominal path following and another setting for when the controller is avoiding an obstacle. In the path following setting, a $Q = 1$ weight is used which ensures accurate following of the reference spline, whereas in the obstacle avoidance setting (when the state is within 5 meters of an obstacle), $Q = 1000$ is used to encourage the vehicle to deviate from the reference if necessary to avoid the obstacle.

These two simple terms embody the non-safety constraint portions of the DDT: Drive close to the desired route, and make efficient progress towards route completion. The rest of the DDT (the safety constraints) is implemented as hard constraints on the NLP and are covered below.

C. *Constraints*

All vehicular systems are non-holonomic and therefore an OCP aiming to control such a system must take the non-holonomic constraints into consideration. These non-holonomic constraints include the system model presented above, as well as further physical constraints like maximum steering angle. There are also constraints enforced for the comfort of the passenger as per the SAE guidelines, which include limits on longitudinal and lateral acceleration as well as jerk. Most importantly, there are the constraints that enforce the feasibility of the predicted vehicle trajectory: The vehicle must stay in free space, defined as inside the road boundaries and not in collision with any obstacle. A complete list of the OCP constraints can be found in Table **??**.

1) Road Boundary Enforcement

There are multiple ways to enforce that a 2D point $[x_X, x_Y]'$, i.e. vehicle position, must be inside a set of two splines. The method we found worked best was a "sidedness" test, enforcing that $[x_X, x_Y]'$ is to the right of the left bound, and to the left of the right bound. The first step in the formulation is to obtain a 2D bound vector $[b_0, b_1]'$ that we enforce the sidedness constraint with respect to. b_0 is calculated as the boundary spline at arclength x_Ξ and b_1 is calculated as the boundary spline at arclength x_Ξ + la where la is a small look-ahead (e.g. 0.01). Then, the sidedness constraint can be enforced as in Eq. 4.

$$\begin{aligned} (x_X - b_0.x) * (b_1.y - b_0.y) - \\ (x_Y - b_0.y) * (b_1.x - b_0.x) > 0 \end{aligned} \tag{4}$$

2) Obstacle Avoidance Enforcement

For each obstacle, and for each step in the MPC horizon, the vehicle's footprint cannot intersect with any obstacle ellipse. Eq. 5 *InEllipse(p, el)* implements the basic operation needed for such a constraint, testing if a 2D point p is inside an ellipsoid el (assuming that the point and the ellipsoid are in the same reference frame, a detail that is worked out in Alg. 4).

Algorithm 4 Obstacle Avoidance

Input: Symbolic set of ellipse trajectories O, symbolic state trajectory S

Output: For each matching O_i, S_i in the horizon, enforce that S_i does not conflict with any object in O_i

```
for O_i ∈ O, S_i ∈ S do
  for O_{i,j} ∈ O_i do
    th = O_{i,j}.theta
    rot = [cos(th), sin(th); -sin(th), cos(th)]
    el = {rot * O_{i,j}.cent, O_{i,j}.rx, O_{i,j}.ry}
    for S_{i,j} ∈ expandFootprint(S_i) do
      ENFORCE(! InEllipse(rot * S_{i,j}.pos, el))
    end for
  end for
end for
```

$$\frac{(p.x - el.x)^2}{el.rx} + \frac{(p.y - el.y)^2}{el.ry} < 1 \tag{5}$$

Now all that is left is to call *InEllipse(p, el)* for each ellipse we want to avoid at each step in the horizon as implemented in Alg. 4. Where the $expandFootprint$ subroutine simply returns 6 points around the border of the vehicle's footprint, and $ENFORCE$ enforces the enclosed constraint in the IPOPT solver. Note that here we are creating $6 * N * M$ constraints. This can be on the order of 100s of constraints for a nominal $N = 15$ and $M < 10$.

IV. MPC SOLUTION

To solve the OCP, nonlinear MPC (NMPC) was used. This decision was due to: (1) The high number of constraints used to enforce non-holonomic system dynamics, comfort, and feasibility, and (2) the nonlinear dynamics of the system.

In order to transform the OCP into a NLP, temporal discretization was applied to the system dynamics over a finite prediction horizon ($N = 15$). Specifically, the Runge–Kutta method (RK4) was used with a time step of $T = 0.25s$. As proposed by Bock and Plitt in [18], multiple shooting was used to enforce the dynamics of the system using constraints, reducing the nonlinearity of the objective function especially in the latter steps of the prediction horizon.

To solve the NLP, the interior point optimizer (IPOPT) [12] from the CasADi [20] software package was used. Computational concerns regarding real-time IPOPT solutions are discussed below. With the NLP solution obtained, receding horizon control is applied.

A. *Real Time Performance*

Real-time performance of the controller is defined as the NLP solver operating faster than the sampling time of the controller (0.25s), i.e. greater than 5Hz. In order to achieve real-time performance, warm-starting the iterative solver with a "decent" solution is essential, allowing the solver to reach an acceptable solution after only a small number of iterations. The warm-start used at time t is normally the solution

Variable(s)	Constraint	Type	Reasoning	Calculation
$\boldsymbol{x}_{k+1}$	$f(\boldsymbol{x}_k, \boldsymbol{u}_k)$	Non-Holonomic	Dynamics of system described via constraints as per multiple shooting [18]	Runge-Kutta (RK4). See Section III-A for system used to model f.
u_δ	$-\frac{\pi}{4} \leq u_\delta \leq \frac{\pi}{4}$	Non-Holonomic	Physical range of wheel shaft	N/A, obtained from fact sheet
x_{v_x}	$0 \leq x_{v_x} \leq v_{MAX}$	Legal	Vehicle cannot exceed speed limit	N/A, supplied by HD Map data
x_{a_y}	$-3.5 \leq x_{a_y} \leq 3.5$	Comfort	SAE Bounds on lateral acceleration	From [19]: $x_{a_y} = \frac{x_{v_x}^2 u_\delta}{l_f + l_r}$
$\dot{x}_{a_x}$	$-10 \leq \dot{x}_{a_x} \leq 15$	Comfort	SAE Bounds on longitudinal jerk	$\dot{x}_{a_x} = \frac{u_a^k - x_{a_x}^{k-1}}{T}$ Where T is the sample time
u_a	$-3.5 \leq u_a \leq 3.5$	Comfort	SAE Bounds on longitudinal acceleration	N/A, input variable
$[x_X, x_Y]'$	$[x_X, x_Y]' \in S^{drive}$	Feasibility	The vehicle must be on the driveable surface	See Sec III-C.1
$[x_X, x_Y]'$	$[x_X, x_Y]' \notin \chi^{occ}$	Feasibility	The vehicle can only occupy free space	See Sec III-C.2
x_Ξ	$0 \leq x_\Xi \leq 1$	Feasibility	Vehicle required to stop at end of route	N/A, state variable

TABLE I: Optimal Control Problem Constraints

obtained by the solver at time $t-1$, following from the assumption that the parameters of the optimization change little from one solve to the next. However, over time this assumption may no longer be true, and the solver may become stuck in a series of warm-started local minima (see Section V-D for an example).

A method of "breaking out" of this local minima is required, a way to search for a more global minima given more IPOPT iterations and a less biased warm-start. Our novel method is to run a parallel solver that consumes the same NLP as the original solver (which we will refer to from now on as the "online" solver), but does not use warm-starting (initial decision variable assignments are all zero) and is allowed 10x the number of IPOPT iterations. We will refer to this new solver as the "exploration" solver.

These two solvers run asynchronously of each other, the online solver is used to control the vehicle as normal, and the exploration controller is used to simply "suggest" new warm-starts that the online solver could use. Whenever the online solver is about to perform a new solve, it polls the exploration solver for its most recent solve. If the polled solution has a lower cost than the cost of the previous online solution, the exploration solution is used for the online solver's warm-start instead of the previous online solution.

The result, viewed from a unified perspective, is a controller than can operate in real-time using a warm-starting scheme but also does not get stuck in local minima (see Section V-D for results).

V. EXPERIMENTS

Experiments were carried out on an AMD 3995WX. The solvers used only a single CPU core and 300MB of RAM.

A. *Reference Line Following*

In this experiment, the only reference we have to follow is the lane center as the ego vehicle completes a right turn at a 4-way intersection. The desired behavior is to have low lateral error from the reference spline, keep the NLP solution time within the 0.25s sampling time, and obey the legal and comfort constraints.

Qualitative results of the controller completing the scenario are shown at:
`https://youtu.be/2Gk3XQKlk38`.

Quantitative results are shown in Fig. 2. As seen in the figure, the lateral deviation stays below 25cm from the reference, well within the lane boundaries. The solution time of the NLP also stays below 0.25s, and the velocity stays close to 8m/s, except when necessary to slow down while taking the right turn to avoid passenger discomfort. The comfort metrics are all within the desired range outlined by SAE.

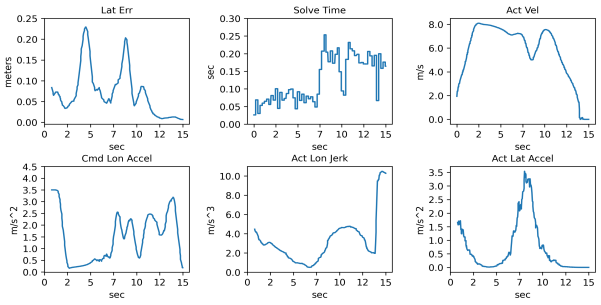

Fig. 2: Quantitative reference line tracking performance.

B. *Static Obstacle Avoidance*

In this experiment, a new reference is introduced: static obstacles. The scenario and desired behavior is the same as in Section V-A, with the addition of a static obstacle course.

Qualitative results of the controller completing the scenario are shown at:
`https://youtu.be/T83wHpZDfd0`.

Quantitative results are shown in Fig. 3. As shown in the top left subplot, the reference (in blue) is tracked well when the controller is not performing an obstacle avoidance maneuver. The controller also successfully avoids the static obstacles (in red). Furthermore, the trajectory stays on the driveable surface (bounds in grey).

C. *Road Boundary*

To show the effectiveness of the road boundary constraints discussed in Section III-C.1, they were removed in this experiment and the resulting vehicle behavior is discussed below. The same static obstacle course testing scenario was used to examine the results.

Qualitative results of the no-road-boundary controller failing to complete the scenario are shown at:
`https://youtu.be/YqpCCIzRw28`.

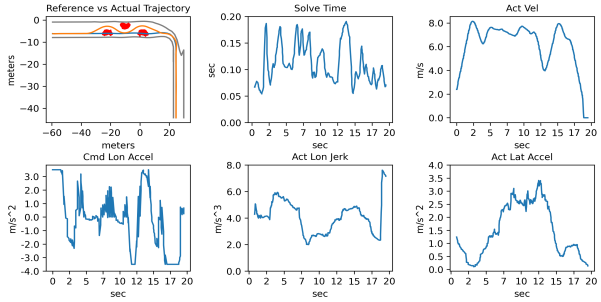

Fig. 3: Quantitative static obstacle avoidance performance.

Quantitative results are shown in Fig. 4. Without the road boundary constraints the controller may arbitrarily decide to avoid the obstacle on either side, possibly violating the road boundary traffic rule (see top left plot). This plot, when compared against the plot in Fig. 3, shows that the road boundary constraints specified in Section III-C.1 are necessary for correct driving behavior.

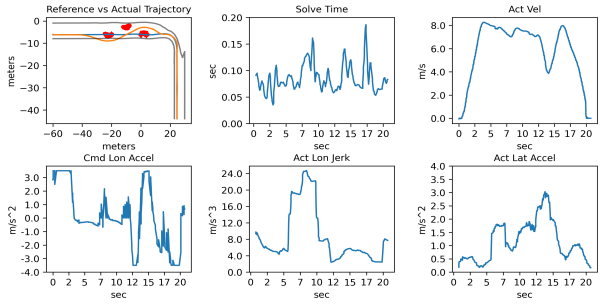

Fig. 4: Quantitative static obstacle avoidance performance of the no-road-boundary setting.

D. *Parallel Solver*

To show the effectiveness of the parallel solver technique discussed in Section IV-A, it was removed (only the online solver is used) in this experiment. The same static obstacle course testing scenario was used to examine the results. The resulting vehicle behavior is discussed below.

Qualitative results of the purely online solver failing to complete the scenario are shown at: `https://youtu.be/cLFCUO3-lMY`. Without the parallel solver the controller gets stuck in a local warm-starting minima and fails to avoid the first obstacle.

Quantitative results are shown in Fig. 5. Without the parallel solver, the controller cannot find a trajectory around the first static obstacle on the course. The failure observed here compared to the success in Section V-B is due to the missing exploration solver discussed in Section IV-A.

E. *Dynamic Obstacle Avoidance*

In this experiment, we introduce a new reference: dynamic obstacles. The scenario used is a straight road with the ego obeying a speed limit of 15m/s. In front of the ego, there is a target vehicle traveling at 7.5m/s. In order to make efficient progress along the route, the ego vehicle should overtake the target vehicle in a comfortable and legal manner.

Qualitative results of the controller completing the scenario are shown at:

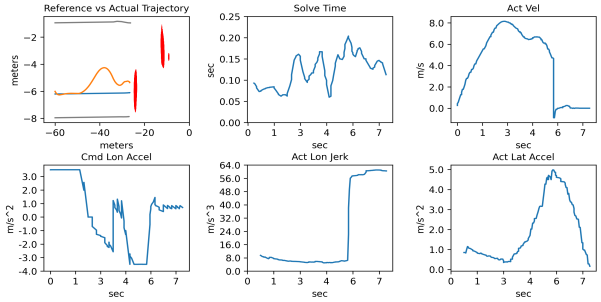

Fig. 5: Quantitative static obstacle avoidance performance of the no-parallel solver setting.

`https://youtu.be/oqjdudBFZ9E`.

Quantitative results are shown in Fig. 6. The reference is tracked well when the controller is not overtaking the target vehicle (see top left plot). The controller also successfully avoids the target vehicle (in semi-transparent red). Furthermore, the controller performs the maneuver efficiently (near 15m/s) and comfortably (SAE guidelines obeyed) while staying on the driveable surface.

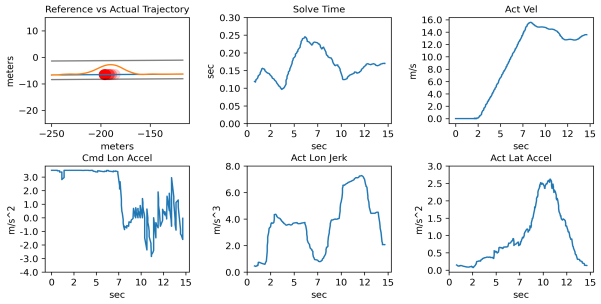

Fig. 6: Quantitative dynamic vehicle avoidance performance.

VI. CONCLUSION

We present a novel OCP formulation and MPC solution to the DDT that allows for unified trajectory planning and control in a feasibility guaranteed manner. The presented scheme is shown to have key advantages over previous DDT motion planning and execution schemes, including direct adherence to SAE passenger comfort guidelines, as well as free space and road boundary conscious control via the OCP constraints. Promising experimental results were achieved for obstacle avoidance, overtaking, and turning maneuvers.

As for future research directions, incorporating learning methods, e.g. [21], for more accurate predictions of vehicle trajectories will enhance the motion planning scheme[1]. Furthermore, we plan to integrate the proposed motion planning and control framework with reinforcement learning based behavioral planners, e.g. [17], for developing feasible decision-making schemes in complex urban environments.

ACKNOWLEDGMENT

This work was supported by NSERC CRD 537104-18, in partnership with General Motors Canada and the SAE AutoDrive Challenge.

[1]Note that the OCP design will not have to change, only the predictive accuracy of the inputs does.

REFERENCES

[1] "Autodrive challenge - autodrive™ challenge." [Online]. Available: https://www.sae.org/attend/student-events/autodrive-challenge

[2] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear mpc algorithm for real-time obstacle avoidance of ground vehicles," in *2013 European Control Conference (ECC)*, 2013, pp. 4136–4141.

[3] U. Rosolia, S. De Bruyne, and A. G. Alleyne, "Autonomous vehicle control: A nonconvex approach for obstacle avoidance," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 469–484, 2017.

[4] K. Berntorp, "Path planning and integrated collision avoidance for autonomous vehicles," in *2017 American Control Conference (ACC)*, 2017, pp. 4023–4028.

[5] Q. Wang, B. Ayalew, and T. Weiskircher, "Predictive maneuver planning for an autonomous vehicle in public highway traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 4, pp. 1303–1315, 2018.

[6] H. Jiang, Z. Wang, Q. Chen, and J. Zhu, "Obstacle avoidance of autonomous vehicles with cqp-based model predictive control," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2016, pp. 001 668–001 673.

[7] H. Guo, C. Shen, H. Zhang, H. Chen, and R. Jia, "Simultaneous trajectory planning and tracking using an mpc method for cyber-physical systems: A case study of obstacle avoidance for an intelligent vehicle," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4273–4283, 2018.

[8] M. A. G. Daoud, "Simultaneous local motion planning and control, adjustable driving behavior, and obstacle representation for autonomous driving," Master's thesis, University of Waterloo, 2020.

[9] S. Li, Z. Li, Z. Yu, B. Zhang, and N. Zhang, "Dynamic trajectory planning and tracking for autonomous vehicle with obstacle avoidance based on model predictive control," *Ieee Access*, vol. 7, pp. 132 074–132 086, 2019.

[10] A. Franco and V. Santos, "Short-term path planning with multiple moving obstacle avoidance based on adaptive mpc," in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2019, pp. 1–7.

[11] R. Dempster, M. Al-Sharman, Y. Jain, J. Li, D. Rayside, and W. Melek, "Drg: A dynamic relation graph for unified prior-online environment modeling in urban autonomous driving," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8054–8060.

[12] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[13] N. Moshtagh, "Minimum volume enclosing ellipsoid," https://www.mathworks.com/matlabcentral/fileexchange/9542-minimum-volume-enclosing-ellipsoid, 2022, [Online; accessed August 16, 2022].

[14] M. J. Todd and E. A. Yıldırım, "On khachiyan's algorithm for the computation of minimum-volume enclosing ellipsoids," *Discrete Applied Mathematics*, vol. 155, no. 13, pp. 1731–1744, 2007.

[15] P. Polack, F. Altché, B. Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" 06 2017, pp. 812–818.

[16] M. A. Daoud, M. W. Mehrez, D. Rayside, and W. W. Melek, "Simultaneous feasible local planning and path-following control for autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[17] M. Al-Sharman, R. Dempster, M. A. Daoud, M. Nasr, D. Rayside, and W. Melek, "Self-Learned Autonomous Driving at Unsignalized Intersections: A Hierarchical Reinforced Learning Approach for Feasible Decision-Making," 9 2022. [Online]. Available: https://www.techrxiv.org/articles/preprint/Self-Learned_Autonomous_Driving_at_Unsignalized_Intersections_A_Hierarchical_Reinforced_Learning_Approach_for_Feasible_Decision-Making/20770486

[18] H. Bock and K. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems*," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984, 9th IFAC World Congress: A Bridge Between Control Science and Technology, Budapest, Hungary, 2-6 July 1984. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667017612059

[19] J. A. Matute, M. Marcano, S. Diaz, and J. Perez, "Experimental validation of a kinematic bicycle model predictive control with lateral acceleration consideration," *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 289–294, 2019.

[20] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[21] N. Deo, E. Wolff, and O. Beijbom, "Multimodal trajectory prediction conditioned on lane-graph traversals," in *Conference on Robot Learning*. PMLR, 2022, pp. 203–212.